

1 CONVOLUTION INTERLEAVER AND DEINTERLEAVER FOR SYSTEMS WITH ERROR
2 CORRECTION ENCODING

3
4 BACKGROUND

5
6 1. Field of the Invention

7 The present invention relates to telecommunications
8 apparatus, systems and methods. More particularly, the present
9 invention relates to convolutional interleavers and deinterleavers
10 of digital modems and transceivers. The invention has particular
11 application to digital subscriber line (DSL) and wireless systems,
12 although it is not limited thereto.

13
14 2. State of the Art

15 In data communications systems, noisy transmission lines can
16 cause data to be corrupted or lost. In order to prevent data loss
17 it is well known to transmit error-checking data along with the
18 transmitted (payload) data. The combination of the payload data
19 and the error-checking data for the payload is often referred to
20 as a codeword. The receiving end of the data transmission can
21 determine if errors have occurred in a particular codeword based
22 on the payload data and error-checking data received. If the
23 amount of error or loss in a codeword is relatively small, the
24 error-checking data can be used to recover the correct payload

1 data.

2
3 Bursts of noise on the transmission medium may cause the data
4 corruption in a codeword to be sufficiently severe such that
5 recovery is not possible. In order to reduce the impact of these
6 bursts of noise, data interleaving techniques are often employed.
7 By interleaving data from different codewords before data
8 transmission and deinterleaving the received data at the receiver,
9 the impact of a burst of noise is spread over a number of
10 different codewords thereby reducing the loss of each codeword to
11 a level where recovery of the payload data in each codeword is
12 possible.

13
14 Well known convolutional interleavers which are used, for
15 example, in ADSL systems, distribute incoming symbols of a
16 codeword according to

$$17 \quad dL(i) = (D-1)*i, \quad i=0,1,2,\dots,(N-1) \quad (1)$$

18 where $dL(i)$ is the delay of the i 'th symbol of a codeword, D is
19 the interleaving depth, and N is the codeword length. Direct
20 implementation of this algorithm in a shift register requires a
21 large memory because the memory size must accommodate the maximum
22 required delay ($\max dL(i)$). The required memory size m_s is defined

$$23 \quad \text{by:} \quad m_s = \max dL(i) = (D-1)*(N-1) \quad (2)$$

24 It will be appreciated by those skilled in the art that the memory

1 size can result in a considerable implementation problem when N
2 and D are large. For example, when using a Reed-Solomon code with
3 codeword length $N=255$ and interleaving depth $D=64$, memories of
4 16,002 8-bit memory cells are required for implementation of the
5 interleaver and deinterleaver. In integrated circuit
6 implementation, large memory blocks require large amounts of "real
7 estate", thereby adding to the cost of the system. Thus, it is
8 desirable to decrease the amount of memory required for
9 implementation of the interleaver and deinterleaver of the system.

10
11 In developing interleaving and deinterleaving algorithms
12 which require less memory, it is useful to determine the minimum
13 number of memory cells required for interleaver and deinterleaver
14 implementation. In finding the minimum number of cells required,
15 it may be assumed that every incoming symbol is written into the
16 cell that is released by reading a current outgoing symbol.

17
18 Using the first memory cell (MC) to delay (store) the second
19 incoming symbol of the codeword, and adding one more MC every time
20 there is not an empty MC among the MCs which have been already
21 taken, the number of MCs required increases up to a minimum number
22 m_{\min} . The minimum m_{\min} is the necessary and sufficient number of
23 memory cells required to implement interleaving and is less than
24 m_s .

The number m_{\min} is equal to the number of incoming symbols which have come before the m_s 'th incoming symbol, but which, on the other hand should be transmitted during or after the m_s 'th incoming symbol. In other words, m_{\min} is equal to a number of incoming symbols with indexes $n \geq 0$, and $n \leq [(N-1)*(D-1)-1]$, which satisfy the inequality

$$n + (D-1)*r \geq (D-1)*(N-1) \quad (3)$$

where

$$r = n - (N*q); \quad q = \text{floor}(n/N) \quad (4)$$

Computer simulation of this algorithm has shown that a number of indexes satisfying inequality (3) is approximately equal to

$$m_{\min} \approx (D-1)*(N-1)/2 \quad (5)$$

Thus, it is seen that the necessary and sufficient number of memory cells for implementing an interleaver is approximately half the number utilized for a standard shift register implementation.

An interleaver/deinterleaver algorithm which can be implemented with a number of memory cells close to m_{\min} is disclosed in U.S. Patent #5,636,224 to Voith et al., entitled "Method and Apparatus for Interleave/De-Interleave Addressing in Data Communication Circuits" which is hereby incorporated by reference herein in its entirety. The Voith et al. algorithm

1 uses parallel circular buffers or FIFOs. The algorithm uses N-1
2 subsets of memory cells (registers), and symbols of a code word
3 are written into separate registers. In reading symbols from
4 registers of the transmitter and writing symbols to registers of
5 the receiver, the Voith et al. algorithm requires computations
6 which include the solving of some specific equation. These
7 computations must be done "on the fly", and therefore require
8 significant computational power. Thus, the Voith et al. algorithm
9 trades off a decrease in memory requirements for an increase in
10 computational power and as a result does not necessarily reduce
11 the expense of the system significantly.

12 SUMMARY OF THE INVENTION

13
14
15 It is therefore an object of the invention to provide an
16 interleaver/deinterleaver which can read/write symbols from/to
17 memory registers without current computations.

18
19 It is another object of the invention to provide an
20 interleaver/deinterleaver which requires a memory which is not
21 much larger than the minimum required.

1 It is a further object of the invention to generate a
2 permutation table which is generated based on a simple computation
3 algorithm.

4
5 In accord with the objects of the invention, an interleaver
6 for a digital modem is provided where consecutive codeword symbols
7 are written into first cells of parallel registers, and reading
8 from registers for interleaving purposes is determined by a
9 permutation table (register) containing N-1 numbers calculated
10 prior to data transmission. The permutation table contains a
11 sequence of N-1 numbers where each number is an index of a
12 register from which the current symbol should be read.

13
14 According to the method of the invention, prior to data
15 transmission, a calculation is made of lengths of the registers
16 (i.e., how many cells each register must have), and a permutation
17 table is determined. Then, during data transmission, the first
18 symbol in each code word is sent directly to the output, the other
19 symbols are sequentially written into the first (input) cells of
20 the corresponding registers, and at the same time, the last
21 (output) cells of the registers are sequentially read in the order
22 determined by the permutation table. After writing the incoming
23 codeword into memory and reading the outgoing codeword, the
24 contents of all memory registers are synchronously shifted by one

1 cell in the output direction.

2
3 A deinterleaver according to the invention corresponds
4 closely to the interleaver of the invention.

5
6 Additional objects and advantages of the invention will
7 become apparent to those skilled in the art upon reference to the
8 detailed description taken in conjunction with the provided
9 figures.

10 BRIEF DESCRIPTION OF THE DRAWINGS

11
12
13 Fig. 1 is a high-level flow/block diagram of a DSL modem;

14
15 Fig. 2 is a high-level block/flow diagram of the interleaving
16 and deinterleaving mechanism of the invention;

17
18 Fig. 3 is a chart illustrating an interleaving algorithm
19 example of the invention; and

20
21 Fig. 4 is a chart illustration a deinterleaving algorithm
22 example of the invention.

1 Figs. 5a and 5b are flow charts representing the method of
2 the invention.

3
4 DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

5
6 The present invention is best understood with reference to a
7 DSL-type modem. Turning to Figure 1, a high level block diagram
8 of a DSL modem 10 is seen. The modem 10 preferably includes a
9 digital interface 20, a transmitter section 30 and a receiver
10 section 40. The transmitter section preferably includes a
11 scrambler 52 which receives data from the digital interface 20, a
12 Reed-Solomon encoder 54, an interleaver 56, a per-carrier bit
13 distributor 58, a mapper 60, a gain element 62, an inverse fast
14 Fourier transform block (IFFT) 64, a cyclic extension block 66, a
15 digital to analog converter 68 and an analog front end transmit
16 block 69 which interfaces with a hybrid 70. The receiver section
17 preferably includes an analog front end receive block 71 which
18 interfaces with the hybrid 70, an analog to digital converter 72,
19 a time equalizer (TEQ) 73, a fast Fourier transform block (FFT)
20 74, a frequency equalizer (FEQ) 76, a demapper 78, a deinterleaver
21 80, a Reed-Solomon decoder 82, and a descrambler 84 which provides
22 data to the digital interface 20. Other than the details of the
23 interleaver 56 and deinterleaver 80, the modem 10 is substantially
24 as would be understood by those skilled in the art. In addition,

1 it will be appreciated by those skilled in the art that the modem
2 10 may be implemented in hardware, software, or a combination
3 thereof.

4
5 Turning now to Fig. 2, the interleaver 56 and deinterleaver
6 80 of the invention are seen in high level format. In particular,
7 the interleaver 56 includes a plurality of parallel registers 100
8 labeled $M_1, M_2, \dots, M_{N-2}, M_{N-1}$, means 102 for writing symbols into the
9 registers, and means 104 for reading symbols out of the registers
10 100 according to a permutation table stored in a permutation
11 register 106. The deinterleaver 80 includes a plurality of
12 parallel registers 150 labeled $MD_0, MD_1, MD_2, \dots, MD_{N-2}, MD_{N-1}$, means
13 152 for writing symbols into the registers according to the
14 permutation table stored in a permutation register 106, and means
15 154 for reading symbols out of the registers 150.

16
17 The permutation table is generated according to a permutation
18 algorithm which includes, prior to data transmission, calculating
19 the lengths of the registers 100 (and 150) and calculating the
20 permutation table. During data transmission, the inputs and
21 outputs of the registers 100 and 150 are switched according to the
22 permutation table. The permutation table contains a sequence of
23 $N-1$ numbers (from 1 to $N-1$), with each number being an index of a
24 register from which the current symbol should be read (in the

1 interleaver) or an index of a register to which the current symbol
2 should be written (in the deinterleaver).

3
4 Fig. 3 illustrates an example of the interleaving algorithm
5 where the codeword length $N = 7$, and the interleaving depth $D = 4$.
6 The first three columns in Fig. 3 represent parameters of symbols
7 which are to be interleaved (the "incoming" symbols), with the
8 first column containing the ordinal numbers (indexes) of the
9 symbols, the second column containing the ordinal numbers of the
10 same symbols within a codeword, and the third column representing
11 symbol delays. The last column of Fig. 3 represents indexes of
12 the outgoing symbols; i.e., the interleaved signals which are
13 being currently transmitted. The "memory states" and
14 "permutations" columns of Fig. 3 demonstrate how the incoming
15 symbols are transformed into the outgoing symbols (for
16 transmission). More particularly, when the first codeword of
17 seven symbols is received, the first symbol (byte 0) is used as
18 the first symbol of the interleaved outgoing signal, and the
19 second through seventh symbols (symbols 1-6) are placed into the
20 first cells of registers M_1 through M_6 . As will be appreciated by
21 those skilled in the art, because symbol 1 is delayed by only
22 three symbols, it appears as the fifth symbol in the first
23 outgoing word; while, because symbols 2-6 are delayed six, nine,
24 twelve, fifteen, and eighteen symbols respectively, they appear in

1 the second, third, and fourth outgoing words. Thus, the first
2 outgoing codeword contains incoming symbols 0 and 1, and five
3 additional symbols (bytes) of stuff (denoted by "x").
4

5 When the second incoming codeword is received, the first
6 symbol (incoming symbol 7) is used as the first symbol of the
7 interleaved outgoing codeword. The second through seventh symbols
8 (symbols 8-13) are placed into registers M_1 through M_6 . Symbol 8
9 is placed into the first and only cell of register M_1 (which is
10 "empty" because symbol 1 which had previously occupied that cell,
11 was written into the first outgoing codeword). Symbols 9-13 are
12 placed into the first cells of registers M_2 through M_6 , the
13 contents of those cells being shifted into the next cells of those
14 registers. The second outgoing codeword is then generated using
15 symbol 7 as the first symbol of that codeword, symbol 2 as the
16 second symbol (symbol 2 having been delayed six symbols), symbol 8
17 as the fifth symbol (symbol 8 having been delayed three symbols),
18 and symbol 3 as the sixth symbol (symbol 3 having been delayed
19 nine symbols). Three additional symbols of stuff are inserted
20 into the third, fourth, and seventh symbol locations of the second
21 codeword.
22

1 When the third incoming codeword is received, the first
2 symbol (incoming symbol 14) is used as the first symbol of the
3 interleaved outgoing codeword. The second through seventh symbols
4 (symbols 15-20) are placed into registers M_1 through M_6 . Symbol 15
5 is placed into the first and only cell of register M_1 (which is
6 "empty" because symbol 8 which had previously occupied that cell,
7 was written into the second outgoing codeword). Symbols 15-20 are
8 placed into the first cells of registers M_2 through M_6 , the
9 contents of those cells being shifted into the next cells of those
10 registers (with the content of the second cells, being shifted
11 into the third cells in the cases of M_4 through M_6). The third
12 outgoing codeword is then generated using symbol 14 as the first
13 symbol of that codeword, symbol 9 as the second symbol (symbol 9
14 having been delayed six symbols), symbol 4 as the third symbol
15 (symbol 4 having been delayed twelve symbols), symbol 15 as the
16 fifth symbol (symbol 15 having been delayed three symbols), symbol
17 10 as the sixth symbol (symbol 10 having been delayed nine
18 symbols), and symbol 5 as the seventh symbol (symbol 5 having been
19 delayed fifteen symbols). An additional symbol of stuff is
20 inserted into the fourth byte location of the third codeword.

21
22 When the fourth incoming codeword is received, the first
23 symbol (incoming symbol 21) is used as the first symbol of the

1 interleaved outgoing codeword. The second through seventh symbols
2 (symbols 22-27) are placed into registers M_1 through M_6 . Symbol 22
3 is placed into the first and only cell of register M_1 (which is
4 "empty" because symbol 15 which had previously occupied that cell,
5 was written into the third outgoing codeword). Symbols 22-27 are
6 placed into the first cells of registers M_2 through M_6 , the
7 contents of those cells being shifted into the next cells of those
8 registers (with the content of the second cells, in the case of M_4
9 and M_5 , being shifted into the third cells, and the content of the
10 third cell, in the case of M_6 , being shifted into a fourth cell).
11 The fourth outgoing codeword is then generated using symbol 21 as
12 the first symbol of that codeword, symbol 16 as the second symbol
13 (symbol 16 having been delayed six symbols), symbol 11 as the
14 third symbol (symbol 11 having been delayed twelve symbols),
15 symbol 6 as the fourth symbol (symbol 6 having been delayed
16 eighteen symbols), symbol 22 as the fifth symbol (symbol 22 having
17 been delayed three symbols), symbol 17 as the sixth symbol (symbol
18 17 having been delayed nine symbols), and symbol 12 as the seventh
19 symbol (symbol 12 having been delayed fifteen symbols).

20
21 When the fifth incoming codeword is received, the process is
22 repeated as indicated. No additional cells are required in

registers M_1 through M_6 as the contents contained in one of the
 cells of each of the registers was written into the previous
 outgoing codeword. Thus, it should be appreciated that in the
 case of the interleaving algorithm where $N=7$, and $D=4$, six
 registers are required, with the first register containing only a
 single cell, the second and third registers containing two cells,
 the fourth and fifth registers containing three cells, and the
 sixth register containing four cells. The number of cells in each
 of the registers is calculated in advance and is not changed
 during data transmission. In addition, the order from which the
 last of the cells of each of the registers is sequentially read is
 repetitive (as seen by comparing the groups of arrows of the
 permutation column of Fig. 3 with respect to each codeword) and
 therefore may be predicted according to a permutation table:

Table 1: Interleaver permutation table for $N=7$, $D=4$

Byte of outgoing codeword	0	1	2	3	4	5	6
Register byte is taken from	Taken directly	2	4	6	1	3	5

1 Thus, the first symbol of every outgoing codeword is taken from
2 the first incoming symbol directly; the second symbol of the
3 outgoing codeword is taken from the second cell of the second
4 register M_2 (having been delayed six symbols); the third symbol of
5 the outgoing codeword is taken from the third cell of the fourth
6 register M_4 (having been delayed twelve symbols); the fourth symbol
7 of the outgoing codeword is taken from the fourth cell of the
8 sixth register M_6 (having been delayed eighteen symbols); the fifth
9 symbol of the outgoing codeword is taken from the first and only
10 cell of the first register M_1 (having been delayed three symbols);
11 the sixth symbol of the outgoing codeword is taken from the second
12 cell of the third register M_3 (having been delayed nine symbols);
13 and the seventh symbol of the outgoing codeword is taken from the
14 third cell of the fifth register M_5 (having been delayed fifteen
15 symbols).

16
17 Once the interleaver permutation table is set, the
18 deinterleaver permutation table is effectively determined. In
19 particular, and as seen in Fig. 4, the symbols of the incoming
20 codeword on the deinterleaving side are distributed to seven
21 parallel registers in an inverse manner according to the following
22 deinterleaver table.

1 Table 2: Deinterleaver permutation table for $N=7$, $D=4$

Byte of interleaved incoming codeword	0	1	2	3	4	5	6
Register to which byte is directed	0	2	4	6	1	3	5

2
3 Thus, symbol 0 is forwarded to register MD_0 ; symbol 1 is
4 forwarded to register MD_2 ; symbol 2 is forwarded to register MD_4 ;
5 symbol 3 is forwarded to register MD_6 ; symbol 4 is forwarded to
6 register MD_1 ; symbol 5 is forwarded to register MD_3 ; and symbol 6
7 is forwarded to register MD_5 . The registers MD_0 through MD_6 are
8 likewise provided with different numbers of cells depending upon
9 how long bytes must be stored before an entire codeword can be
10 reconstructed. As seen in the memory states column of Fig. 4, the
11 number of cells for registers MD_1 through MD_6 is essentially the
12 reverse of the number of cells for registers M_1 through M_6 of the
13 interleaver, and the extra register (M_0) is provided with four
14 cells.

1
 2 With the register structure provided, and utilizing the
 3 deinterleaving table, the codewords are regenerated by taking the
 4 symbols located in the last cells of each of the registers in
 5 order. As shown in Fig. 4, with $N=7$ and $D=4$, the first
 6 regenerated symbol occurs after receiving the fourth incoming
 7 codeword. Thereafter, all cells of all seven registers are
 8 utilized with symbols being written into the first cells of each
 9 of the registers, and symbols already sitting in cells either
 10 being shifted into adjacent cells or being written out to
 11 regenerate a codeword.

12
 13 Because the interleaver and deinterleaver use a simple table,
 14 complex computations are not required to implement them. In
 15 addition, the interleaver and deinterleaver of the invention do
 16 not require memories which are significantly larger than the
 17 minimum required. In fact, in the general case (for odd N , and
 18 any value for D), the total number of interleaver memory cells m_I
 19 is equal to

$$m_I = (N-1) * [((D-1)/2) + 1] \quad (6)$$

21 which is close to the minimum m_{\min} and almost two time less than m_s .

22 If $N=255$ and $D=64$, then $m_I = 8255$, $m_{\min} = 8001$, and $m_s = 16002$.

23 Similarly, the total number of interleaver memory cells

1 $m_D = m_I + D$. Thus, for the same example where $N=255$ and $D=64$, $m_D =$
 2 8319.

3
 4 Given all of the above, methods of interleaving and
 5 deinterleaving according to the invention can be described
 6 according to initialization steps and processing steps. The steps
 7 utilize the following designations:

8 D - interleaving depth;

9 N - code word length;

10 M_j - j 'th interleaver register;

11 L_j - j 'th interleaver register length;

12 MD_j - j 'th deinterleaver register;

13 LD_j - j 'th deinterleaver register length;

14 P - permutation register with length $N-1$;

15 $S_{in}(i)$ - i 'th incoming symbol of a codeword;

16 ceil - rounding up to the next whole number;

17 floor - rounding down to the next whole number.

18
 19 Referring now to Fig. 5a, according to a method of the
 20 invention, in a first initialization step for interleaving, the
 21 length L_j of the j 'th register M_j is determined at 200 according
 22 to:

$$L_j = \text{ceil}(j \cdot D / N), \quad j=1, 2, \dots, (N-1) \quad (7)$$

In a second initialization step for interleaving, at 205, a permutation register holding indications of the interleaving permutation table is initialized. Initialization can be represented by calculating

$$f(n) = n_{\text{Mod}N} \quad (8)$$

$$k(n) = (D-1) \cdot f(n) + n \quad (9)$$

$$m(n) = k(n) - N \cdot (D-1) \quad (10)$$

where $n=0, 1, 2, \dots, (N \cdot D - 1)$,
and by setting indication $f(n)$ into the $m(n)$ 'th cell of the permutation register P when $N \cdot D > k(n) > N \cdot (D-1)$.

Initialization may also be represented by the following code:

```

for n=0:1:(N*D - 1)
    f=n-N*floor(n/N);
    k=(D-1)*f + n;
    if k>N*(D-1) and k<N*D
        m=k-N*(D-1);
        P(m) = f;
    end
end

```

The results of the above code for $N=7$ and $D=4$ are illustrated in Appendix 1 below:

1

Appendix 1

2

n	$f=n-7*\text{floor}(n/7)$	$k=n+3*f$	If $k>21$ and $k<28$ $m= k-21$	$P(m)=f$
0	0	0	-	-
1	1	4	-	-
2	2	8	-	-
3	3	12	-	-
4	4	16	-	-
5	5	20	-	-
6	6	24	3	$P(3)=6$
7	0	7	-	-
8	1	11	-	-
9	2	15	-	-
10	3	19	-	-
11	4	23	2	$P(2)=4$
12	5	27	6	$P(6)=5$
13	6	31	-	-
14	0	14	-	-
15	1	18	-	-
16	2	22	1	$P(1)=2$
17	3	26	5	$P(5)=3$
18	4	30	-	-
19	5	34	-	-
20	6	38	-	-
21	0	21	-	-
22	1	25	4	$P(4)=1$
23	2	29	-	-
24	3	33	-	-
25	4	37	-	-
26	5	41	-	-
27	6	45	-	-

In a third initialization step for interleaving, at 210, $N-1$ registers M_j are initialized with lengths L_j . Memory set M is an association of M_j such that $M = [M_1, M_2 \dots M_{N-1}]$.

After initialization of the registers M_j and the permutation register P , codewords having incoming symbols $S_{in}(i)$ (with indexes $i=0,1,2,\dots,N-1$ where $i=0$ corresponds to the first element of a codeword) are processed (interleaved) as follows. First, a determination is made at 215 whether or not $i=0$. If $i=0$, at 220, the current outgoing symbol is set equal to a current incoming symbol; i.e.,

$$S_{out} = S_{in}(0) \quad (11)$$

Then, at 225 the contents of all registers are shifted by one cell towards the register output; i.e.,

$$M_j(m_j) = M_j(m_j-1), \quad m_j = 2,3,\dots,L_j; \quad j=1,2,\dots,N-1 \quad (12)$$

On the other hand, if at 215 $i>0$, the current incoming symbol $S_{in}(i)$ is written at 230 in the now vacant first memory cell of the i 'th register M_i ; i.e.,

$$M_i(1) = S_{in}(i) \quad (13)$$

and at 235 a current outgoing symbol is read from the shifted out last memory cell of the $P(i)$ 'th register $M_{P(i)}$; i.e.,

$$S_{out} = M_{P(i)}(L_{P(i)}) \quad (14)$$

After step 225 or step 235, the index i is incremented at 240, and a check is made at 245 as to whether $i=N$. If $i=N$, at 248, i is reset to zero, and the loop continues at step 215. If $i \neq N$, i is not reset, but the loop is continued at step 215.

The initialization steps and processing steps for deinterleaving correspond closely to the initialization and processing steps discussed above with respect to interleaving. As seen in Fig. 5b, a first initialization step 250 for deinterleaving involves calculating register lengths LD_j which represent the number of memory cells in each of the registers MD_j according to:

$$LD_j = D - \text{floor}(j \cdot D/N), \quad j=0,1,2,\dots,(N-1) \quad (15)$$

It should be noted that when parameters N and D are equal in both the upstream and downstream directions, LD_j may be also found by reordering the L_j according to

$$LD_0 = D \quad (16)$$

$$LD_j = L_{N-j} \quad j=1,2,\dots,(N-1) \quad (17)$$

A second initialization step 255 for deinterleaving involves initializing the permutation register $P(j)$ which is identical to

1 the interleaving initialization of the permutation register,
 2 except that $P(0)$ is set equal to 0, such that the first incoming
 3 symbol of the incoming interleaved codeword is placed into the
 4 first register of the deinterleaving registers.

5
 6 A third initialization step 260 for deinterleaving involves
 7 initializing N registers MD_j with lengths LD_j . The full memory set
 8 MD is an association of MD_j :

$$MD = [MD_0, MD_1 \dots MD_{N-1}] \quad (18)$$

9
 10
 11 After initialization of the registers MD_j and the permutation
 12 register P , codewords having incoming symbols $S_{in}(i)$ (with indexes
 13 $i=0,1,2,\dots,N-1$ where $i=0$ corresponds to the first element of a
 14 codeword) are processed (deinterleaved) as follows. First, at
 15 265, the current incoming symbol $S_{in}(i)$ is written in the first
 16 memory cell of the $P(i)$ 'th register $MD_{P(i)}$ according to:

$$MD_{P(i)}(1) = S_{in}(i) \quad (20)$$

17
 18 Then, at 270, a current outgoing symbol S_{out} is read from the last
 19 memory cell of the i 'th register MD_i ; i.e.,

$$S_{out} = MD_i(L_{P(i)}) \quad (21)$$

20
 21 At 275, i is incremented, and at 280 a determination is made as to

1 whether $i = N$. If $i = N$, at 285 the contents of all registers are
 2 shifted by one cell towards the register output; i.e.,

$$3 \quad MD_j(m_j) = MD_j(m_j-1), \quad m_j = 2, 3, \dots, L_j; \quad j=0, 1, 2, \dots, N-1 \quad (22)$$

4 and, at 290, i is reset to zero. The loop then continues at 265.

5 If at 280, on the other hand, $i \neq N$, the loop continues at 265

6 with symbols being written into memory cells.

7
 8 A MATLAB® simulation program for the interleaver
 9 initialization and processing and the deinterleaver initialization
 10 and processing is attached as Appendix 2 below:

11 Appendix 2 - Interleaver/Deinterleaver Simulation in Matlab

```

12 N=input('enter Code Length N ');
D=input('enter Interleaving Depth D ');
%message generation
M=D+5;                                     %example: for algorithm checking
Bin=[0 1:(M*N-1)];                         %example: Incoming Data
%INTERLEAVER
%Registers Size Calculation
Sz=zeros(size(1:(N-1)));
for qq=1:(N-1)
    Sz(qq)=ceil(qq*D/N);
end
Sz;                                         %demo:Interleaver Registers Size
MemoryI=sum(Sz);                           %demo:Number of Interleaver Memory Cells
%Registers initialization
R=zeros((N-1),D);                         %In real implementation: Ri=zeros(size(1:(Sz(i))))
%Calculation of the Permutation Table
P=zeros(size(1:N));
for q=1:N*D
    qm=q-1;
    ff=floor(qm/N); f=qm-N*ff;
    qk=(D-1)*f+qm;
    if qk>N*(D-1) & qk<N*D
        m=qk-N*(D-1);
        P(m)=f;
    end
end
end
P';                                         %demo: The Permutation Table

```


1

```

%Interleaver Processing
for n=1:(N*M)
    nn=n-1; ii=floor(nn/N); i=nn-N*ii;
    if i==0
        Bout=Bin(n);
        z=2:1:D; R(:, z)=R(:, z-1);
    else
        R(i,1)=Bin(n);
        Bout=R(P(i), Sz(P(i)));
    end
    XByout(n)=Bout;
    DBin(n)=Bout;
end
XByout';
%DEINTERLEAVER
%Register Sizes Calculation
Sz=[D zeros(size(1:(N-1)))];
for qq=1:(N-1)
    Sz(N+1-qq)=ceil(qq*D/N);
end
Sz;
MemoryD=sum(Sz);
%Registers initialization
R=zeros(N,D);
P=P+1;
%Deinterleaver Processing
for n=1:(N*M)
    nn=n-1; ii=floor(nn/N); i=nn-N*ii; j=i+1;
    if i==0
        R(1,1)=DBin(n);
    else
        R(P(i),1)=DBin(n);
    end
    DBout=R(j, Sz(j));
    if j==N
        z=2:1:D; R(:, z)=R(:, z-1);
    end
    DByout(n)=DBout;
end
DByout';
MemoryI
MemoryD

```

1 There has been described and illustrated herein a preferred
2 embodiment of an interleaver and a deinterleaver for a digital
3 modem. While a preferred embodiment of the invention has been
4 described, it is not intended that the invention be limited
5 thereto, as it is intended that the invention be as broad in scope
6 as the art will allow and that the specification be read likewise.
7 Thus, while the invention was described with reference to symbols
8 which are each a byte long, it will be appreciated that the
9 symbols could be of other lengths. Also, while particular code
10 has been listed for initializing the permutation register, it will
11 be appreciated that other code could be utilized. Likewise, while
12 particular Matlab code has been provided for implementing
13 interleaver and deinterleaver initialization and processing, other
14 code could be utilized. Further, while the invention has been
15 described with reference to "registers", it will be appreciated by
16 those skilled in the art that the term "register" is intended to
17 be broadly understood to include all different types of storage
18 elements including FIFOs, shift-registers, circular buffers, RAM,
19 etc. Indeed, it should be noted that where the invention is
20 implemented with a well-known circular buffer rather than with
21 shift registers, a physical shift of data (as suggested in Figs.
22 5a and 5b) is not required. Instead, the equivalent is
23 accomplished through the adjustment of read and write pointers.
24 It will also be appreciated that the registers, read means, and

1 write means of the invention may be implemented in hardware,
2 software, or a combination thereof. In addition, while the
3 permutation register of the invention has been described as
4 holding N-1 cells, it will be appreciated that the permutation
5 register can include an additional cell (cell 0) which is set to
6 value 0. Also, while the size of each cell of the permutation
7 registers was not specified, it will be appreciated that the
8 permutation register cells can be byte-wide for convenience, or
9 may be sized simply to accommodate the number of symbols in the
10 codewords; i.e., if the codeword has seven bytes, only three bits
11 are needed in each permutation register cell to identify each
12 register, whereas if the codeword has 128 bytes, seven bits would
13 be needed.

14
15 It will further be appreciated by those skilled in the art
16 that while the invention was described as using the permutation
17 register for reading data out of the interleaver, in fact, by
18 rearranging the cell depths of the respective registers, the
19 permutation register may be utilized for writing data into the
20 registers. Thus, the first incoming symbol (symbol 0) will
21 continue to be directly read out. Using the permutation register,
22 the second incoming symbol is then written to the fourth register
23 M_4 (which is provided with only a single cell), the third incoming
24 symbol is written to the first register M_1 (which is provided with

1 two cells), the fourth incoming symbol is written to register M_5
2 (which is provided with two cells), the fifth incoming symbol is
3 written to M_2 (which is provided with three cells), the sixth
4 incoming symbol is written to M_6 (which is provided with three
5 cells), and the seventh incoming symbol is written to M_3 (which is
6 provided with four cells). Reading of the symbols out of the
7 registers may then be conducted in direct order; i.e., after
8 reading out symbol 0 directly, registers M_1 through M_6 are read in
9 sequential order. In a similar fashion, instead of using the
10 permutation register to write data into the deinterleaver, the
11 permutation register may be used to read data out of the
12 deinterleaver. Where the permutation register is used in that
13 manner, interleaved symbols may be read in sequential order into
14 the deinterleaver registers which are provided with appropriate
15 depths.

16
17 It should be appreciated that the contents of the permutation
18 register used to write data into the interleaver (instead of
19 reading data out of the interleaver) may be derived from the
20 contents of the permutation register which reads data out of the
21 interleaver. This is simply accomplished by causing the register
22 contents (i.e., register # for reading data) to be used as the
23 register address (i.e., outgoing codeword symbol #), and vice

versa. Thus, for example, in Table 1, where the contents of symbol 1 is taken from register 2 (for a permutation register used to read data out of the interleaver), symbol 2 would be taken from register 1 (for the permutation register used to write data into the interleaver). Likewise, in Table 1, where the contents of symbol 2 are taken from register 4, in the permutation register used to write data into the interleaver, the contents of symbol 4 are taken from register 2. As a result, Table 1 can be extended as follows:

Outgoing codeword symbol #	0	1	2	3	4	5	6
Register # for reading data	Taken directly	2	4	6	1	3	5
Register # for writing data	Taken directly	4	1	5	2	6	3

and the permutation register can be taken from either row 2 or row 3 of the above table depending upon whether the permutation register is being used to read data out of registers or write data into registers. It should be noted that Table 2 may also be similarly extended.

1 Those skilled in the art will also appreciate that while
2 Figs. 5a and 5b suggest that for every symbol written into the
3 interleaver or deinterleaver, another symbol is read out, the
4 order may be different. For example, an entire codeword may be
5 written in and then an entire codeword read out.

6
7 It will therefore be appreciated by those skilled in the art
8 that yet other modifications could be made to the provided
9 invention without deviating from its spirit and scope as so
10 claimed.